(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－31096

(43)公開日　平成11年(1999)２月２日

(51)Int.Cl.[6]		識別記号		F I		
Ｇ０６Ｆ	12/00	５２０	Ｇ０６Ｆ	12/00	５２０Ａ
	17/30			15/413	３１０Ｚ

審査請求　未請求　請求項の数２　書面　（全 ９ 頁）

(21)出願番号　特願平9－219700

(22)出願日　平成９年(1997)７月11日

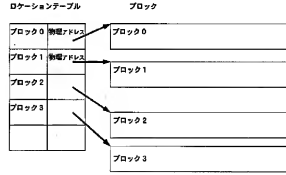
(71)出願人　593050596
アネックスシステムズ株式会社
東京都港区北青山３丁目７番１号

(71)出願人　592159324
玉津　雅晴
東京都八王子市南大沢４丁目19番15号１

(72)発明者　玉津　雅晴
東京都八王子市南大沢４－19－15－１

(54)【発明の名称】　データ格納検索方式

(57)【要約】
【目的】従来のデータ格納方式は、ブロックの物理的な位置の連続性や、格納する為のブロックを事前に作成しておく必要があるなど制約が多い物であった。また、検索にはインデックスの作成が必要で、創生の時間が長く、また、更新処理を行なう場合には、インデックスの更新が発生し排他の範囲が広くなる為、デッドロックが発生しやすい構造を持っていた。これを半導体のランダムアクセス性を利用し、高速化と同時にメンテナンスの負荷を最小限にする。
【構成】インデックスの代わりとして、ロケーションテーブルと代替キーテーブルとよぶものを新たに導入する。またレコードは原則として複数のレコードを１つのブロックに格納する。可変長レコードやスパンドレコードも扱う事が可能である。ロケーションテーブルは、格納ブロックを管理する。代替キーブロックは代替キーとブロック番号、主キー値からなり、いずれもこのテーブルを検索する事により目的のレコードの検索を行なう。テーブルの検索は、バイナリー・サーチが良く知られている高速な方法であるが、他の方法でも可能である。



1

【特許請求の範囲】

【請求項１】コンピューターにおける、データ格納方式において、

①データレコード中に一つのユニークなキー（異なるレコードのキー値が重複しないもので、以降、主キーと呼ぶ）と、ゼロ個もしくは１個以上のノンユニークなキー（異なるレコードのキー値が重複してもかまわないもので、以降、代替キーと呼ぶ）を持つレコードを固定長のブロックの中に主キーの順番に並ぶように１個以上格納し、

②ブロックはプライマリーブロックとオーバーフローブロックで構成し、レコードはまずプライマリーブロックに格納し、

③レコードの挿入によりプライマリーブロック中に格納できなくなった場合に、そのプライマリーブロックに対してオーバーフローブロックを割り当て、１つのオーバーフローブロックでは格納できない場合は更に１つづつオーバーフローブロックを割り当て、各々のブロックを連携してレコードを格納し、

④レコードの追加に対して最終プライマリーブロックに格納できない場合は、新たなプライマリーブロックを割り当ててレコードを格納し、

⑤プライマリーブロックの位置管理は、ロケーションテーブルを用いることにより、各々のブロックの物理的な位置に関しては何らの制限を受けず配置でき、

⑥各々のブロックは予め作成しておく必要が無く、必要に応じて作成し、物理的なデータ格納エリアが満杯になるまで作成でき、

⑦複数の特定のプライマリーキーの後に、レコード挿入が多数行われる型のファイルに対しては、挿入が行われる位置で複数のサブレンジに分割し、挿入ではなくレコード追加とし、オーバーフローレコードの発生を防ぎ、

⑧プライマリーおよびオーバーフローブロックを複数のコンピューター上に分割して保持できる構造の、格納方式。

【請求項２】前記の格納方式において、

①キーによる検索の為の方法としてインデックスを用いず、ロケーションテーブルと代替キーテーブルを使用し、各々は、格納するレコードの数に見合った大きさの連続領域を予め確保する。

②ロケーションテーブルはロケーションテーブルの番号（０からの連続番号でプライマリーブロックの番号と同一）とプライマリーブロックの物理アドレスを一つのエントリーにし、ロケーションテーブルの番号の順番に並べた構造とする。主キーによる検索は、ロケーションテーブルに対して検索を行う。目的のキー値を含むブロックをもとめ、次にブロック中のレコードを検索する事により目的のレコードの検索を行なう。

③レコードの格納にランダムアクセス型記憶媒体装置を使用する場合は、ロケーションテーブルのエントリーに該当ブロックの主キーを含むものを１エントリーとし、ブロック検索が、ロケーションテーブルのみで行なえるようにする。

④代替キーを使用する場合は、代替キーテーブルを作成する。代替キーテーブルは代替キーブロックに格納する方式とし、そのブロックを代替キーブロックと呼ぶ。代替キーブロックは予め同一の大きさで必要数を連続して確保し、一つの代替キーブロックに複数のエントリーが格納可能とする。エントリーは、代替キーと該当レコードが格納されているブロック番号、該当レコードの主キーからなり、これを代替キーの昇順に並べたもので、同一の代替キーに複数のレコードが存在する場合は、これらを一つのエントリーとする方法も使用できる。同一の代替キーを持つエントリーは同一の代替キーブロックに格納する。同一の代替キーを持つエントリーの数が多い、もしくは代替キーの挿入により、代替キーブロックに格納できない場合代替キー・オーバーフロー・ブロックを代替キーブロックに追加し格納する。

⑤代替キーによる検索は、この代替キーブロックを検索することにより、目的の代替キー値を持つエントリーを検出し、そのエントリーから目的のレコードの格納されているブロック番号を知り、そのブロック中の当該レコードの検索を行う。代替キーブロックのみを検索の対象とし、探索された代替キーブロックに代替キー・オーバーフロー・ブロックがある場合は、両方のブロックに格納されているエントリーを検索する。

⑥データレコードが最終格納レコード数に比較して非常に少ない場合は、その後のレコードの追加で、代替キーの挿入が頻繁に発生する可能性が大きいが、この場合は、最終レコード数に対応した代替キーブロックに最初から格納せず、レコード数が代替キーブロックの数に到達するまで、プレ代替キーブロックに格納する。プレ代替キーブロックは、代替キー・ブロックと同様な構造とし、ブロックの数は代替キーブロックの数のエントリーが格納できる大きさを、ブロックのサイズで割って求めた数とする。プレ代替キーブロックのエントリー数が、代替キーブロックの数と同じになった時点で、プレ代替キーブロックから代替キーブロックにエントリーを移動する。移動時には、代替キーブロックには原則として一つのエントリーを格納するが、同一の代替キーを持つエントリーは、同一の代替キーブロックに格納する。同一の代替キーを持つエントリーの数が多くて代替キーブロックに格納できない場合は、代替キー・オーバーフロー・ブロックを追加し格納する。

⑦主キーによる、順次読み出しは、ロケーションテーブルでプライマリー物理ブロックアドレスを検出し、そのプライマリーブロック内のレコードと、オーバーフローブロックが存在する場合にはオーバーフローブロック内のレコードを順次読み出すことにより読み出しを行う。次のブロックはロケーションテーブルの次のエントリー

を求め、そのエントリーから物理ブロックを求める。

⑱代替キーによる順次呼び出しは、まず、⑮により最初のレコードを読み出し、次のレコードの読み出しは、代替キーブロックの次のエントリーのレコードを順次に呼び出す事により行なう。検索方式。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明はコンピューターのデータ格納、読み出しに関連し、高速かつ高格納率でデータの格納及び検索が行なえ、メンテナンスの手間を大幅に削減するもので、データ管理に関する。

【０００２】

【従来の技術】従来のデータ格納方式は、ブロックの物理的な位置が連続していることや、予め格納する為のブロックを作成しておかなければならないなどという、使用する上で制約が多い物であった。また、ランダムアクセスに関しては、一部の方式を除くと、インデックスの作成が必要で、創生の時間が長く、また、更新処理を行なう場合には、インデックスの更新が発生し排他の範囲が広くなる為、デッドロックが発生しやすい構造を持っていた。インデックスを使用しない方式としてダイレクトアクセス方式があるが、この方式は、レコードのキーと格納場所をランダマイズルーチンという特殊なプログラムで関連付ける方式であり、順次アクセスが実用上無理であり、格納効率もインデックスを使用する方法に比較すると、低いものであった。

【０００３】

【発明が解決しようとする課題】このような方式は、磁気ディスクを前提にした技術であったため、止むを得ない面があった。一方、半導体の価格低下はめざましいものが有り、半導体を記憶装置として利用する条件が整いつつある。半導体は、物理的な移動や回転といった動作が不要であり、アドレスが連続していなくても高速な格納、読み出しが可能となる。この性質を利用し、半導体のみまたは半導体とランダムアクセス型記憶媒体を用いて記憶装置を構成すれば高速な格納・検索処理が行なえる。半導体は主記憶装置を使用する事も可能であるし、本発明用に外部記憶装置として構成する事も可能である。データの格納はブロックに順次に格納し、データの挿入によりあふれた場合はオーバーフローブロックを利用して格納し、ランダムアクセスに関しては、インデックスを使用せず格納ブロックを管理するロケーションテーブルまたは代替キーブロックを使用し、それに対して検索を行なう事により、高速な格納、読出しを可能とし、データの格納効率も良くデッドロックの発生を最小限にとどめる。

【０００４】

【課題を解決する為の手段】インデックスの代わりとして、ロケーションテーブルと代替キーテーブルとよぶものを新たに導入する。また、レコードは単独で格納するのではなく、原則として複数のレコードを１つのブロックに格納する。ブロックの構成は図４に示す。この図中のＦＲＯＭ、ＴＯはそれぞれ当該ブロック中の最小キー値、最大キー値を示すが、必ずしも両方が必要ではなく、片方のみでも本方式の適用は可能である。長さが長いレコードを使用する場合は、一つのブロックに一つのレコードとしてもよい。また、２つ以上のブロックに１つのレコードが格納される形態（スパンドレコード）も可能である。また、ブロックはプライマリーブロック、オーバーフローブロックそれぞれ毎に、１つのファイルでは全て同一の長さとし、領域管理を容易にする。ブロックに格納することにより、可変長レコードを格納しても、ブロック単位では固定長となるため、ブロック検出は固定長の操作で行なえる。ロケーションテーブルは事前に必要な連続領域を確保する。ロケーションテーブルの１レコード（エントリー）は、データレコード格納領域のプライマリーブロック１ブロックを管理する。複数のプライマリーブロックを管理する方法も可能であるが、プライマリーブロックのサイズは任意の大きさがとれるので、大きさの変更を行なえばよく複数ブロックの管理は行なわない方が管理が容易になる。ブロックはプライマリーとオーバーフローに分けられ、ロケーションテーブルで管理されるのはプライマリーブロックのみである。レコードはまずプライマリーブロックに格納される。そのブロックに対して、レコードの挿入が発生しそのブロックに格納できなくなった場合は、そのブロックに対するオーバーフローブロックを１つ割り当てる。そのオーバーフローブロックにも格納出来なくなった場合は更に１つのオーバーフローブロックを割り当てる。オーバーフローブロックはプライマリーブロックの従属ブロックとして管理され、プライマリーブロックからポインティングされるのみで、ロケーションテーブルには管理されない。

【０００５】オーバーフローブロックをロケーションテーブルで管理しないので、ロケーションテーブルへのレコード挿入が発生せず、ロケーションテーブルの書換えに要する時間が必要最小限ですみ、更に、ロケーションテーブルの書換えが１レコードのみである為、排他が発生してもその範囲が極小化でき、デッドロックの可能性が大幅に減少する。デッドロックは、１つのコンピューター上で、２つの異なるタスクが、２つ以上の同じ資源を異なる順序で排他を行なう為に発生し、排他の順序が同一であれば発生せず、排他の範囲が狭ければ確率は減少する。従来のインデックスは、数レベルから成り立っている。必要なレコードをアクセスし、更新処理を行った際にレコードのキーが変更されると、インデックスも変更されるが、これが最下位のインデックスにのみ影響する場合は排他範囲は限定されるが、場合によっては上位のインデックスに影響する場合もある。そうなると、多数のレコードの範囲に排他が及ぶとともに、インデッ

クスの更新にも時間を要する事から、排他の時間が長時間に及ぶことになり、デッドロックが多発する原因となる。

【０００６】

【作用】コンピューターの記憶領域に本方式を適用する。

【０００７】

【実施例】本方式で格納するレコードは、必ず１つのユニークな主キーとゼロ個若しくは１個以上のノンユニーク（結果的にユニークであっても問題はない）なキー（代替キー）を持つレコードに限定される。ユニークなキーを持たないレコードは対象としない。但し、ユニークなキーが無いレコードに対して、レコードの追加時に連続番号などのユニークなキーを強制的に付加し、読み出しは物理順もしくは代替キーの順にのみ行なう事は可能である。この場合は挿入が行われない為、レコードの更新によりレコード長が増加する以外にはオーバーフローブロックが発生しない。以下の説明で追加とは、現在格納されているレコードの主キーより大きな主キーを持つレコードの格納をさし、挿入は、最大の主キーより小さな主キーを持つレコードの格納をさす。

【０００８】まず、格納方式を説明する。ロケーションテーブルは予めレコードの格納予定数とブロックの大きさ、ロケーションテーブル１レコード当たりのプライマリーブロック数から計算して必要十分な領域を連続領域として確保しておく。同様に代替キー・ブロックも格納するレコードの数のエントリーが格納できる様に、ブロックの大きさと数を決定し、連続領域として確保する。しかしながら、当初想定した数を上回って格納が行われる場合に、連続領域が満杯になり、格納が不可能になってしまう可能性がある。このような場合には、更に追加で連続領域の確保を行ない、アドレス変換テーブルを用いて、複数の連続領域があたかも１つの連続領域であるかのように扱う事により、当初の想定レコード数を超える数の格納が行われるような事態にも対処が可能である。代替キーが複数ある場合は、それぞれ毎に領域を確保する。異なる代替キーの代替キーブロック同士は連続する必要は無い。ロケーションテーブルとブロックの関係を図２に示す。サブレンジに分割する格納方式を採用する場合は、各々のサブレンジの予定格納レコード数に見合ったロケーションテーブルをサブレンジ毎に作成する。ロケーションテーブルの各々は連続している領域である必要があるが、ロケーションテーブル同士は連続した領域である必要はない。代替キーブロックは、すべてのレコードを対象とした大きさの連続領域を、分割せずに用意する。

【０００９】最初のレコードを格納する場合は、まず、最終ポインターを排他モードで参照する。最終ポインターは、ブロックとロケーションテーブルをどこまで使用しているかを管理するもので、形式は図１に示すとおり。最初は格納されているレコードが何も無いので、まず、最終ポインターに１ブロック目が最後のブロックである事を登録する。最終ポインターにはブロック＃と主キーの値とを登録する。次に１つのプライマリーブロックを排他モードで確保し、その物理アドレスとブロック番号（この場合は０。番号は０からスタートする。）を登録する。ブロックをディスク上に確保する場合は、主キーの値もエントリーに含める。次にそのブロックにレコードを登録する。その後総ての排他を解除する。

【００１０】２レコード目の登録は、まず、最終ポインターを排他モードで参照し、主キーが最終ポインターのキー値より大きいか否かを判定する。まず、レコードの追加の場合として説明を行なう。ブロック番号が０なのでロケーションテーブルのブロック番号０を排他モードで参照し、０ブロックの物理位置を知り、その物理位置にあるブロック０を読み出す。そのブロックに十分な空き領域があれば、レコードを格納し、最終ポインターに主キーの登録を行ない、総ての排他を解除する。

【００１１】以下、同様の操作で追加格納が行われるが、０ブロックに入りきらない場合の説明を行なう。ｍ番目のレコード追加で上記操作を行ない０ブロックを読み出した際に、０ブロックに十分な余裕が無い場合、プライマリーブロックを１つ（ブロック番号１）排他モードで確保する。ｍ番目のレコードをブロック１に格納する。その後、ロケーションテーブルの２レコード目を排他モードで参照し、このレコードにプライマリーブロック１の物理位置を登録する。この後総ての排他を解除する。このように、追加は論理的に最後のレコードの後になる位置に格納される。サブレンジに分割されている場合は、サブレンジ毎に同様の操作が発生する。

【００１２】次に、レコードの挿入になる場合を説明する。既に複数のロケーションテーブルレコード、プライマリーブロック、データレコードが存在しているとする。まず、挿入されるレコードを、どのブロックに格納すべきかを判別する必要がある。これはロケーションテーブルに対して検索を行なう。高速な検索の方法の一例としてバイナリー・サーチが知られている。ここではバイナリー・サーチを行なう方法を例として記述するが、他の方法でも目的のエントリーを探す事は可能である。方法としては２分割点を探し、そのレコードが指しているプライマリーブロックにオーバーフローブロックがある場合にはオーバーフローブロックを含めて、そのブロック（当該ブロック）に格納されているレコードの主キーの値（以下、格納主キー値と略す）と挿入レコードの主キーの値（以下、挿入主キー値と略す）を比較する。挿入主キー値が格納主キー値の最小値より大きいか、当該ブロックの格納主キー値の最大値より大きく、次のブロックの格納主キー値の最小値より小さい場合は、当該ブロックに格納される。この判定に該当しない場合は、更に当該ブロックの格納主キー値と挿入ブロック主キー

値の大小を比較し、挿入主キー値が小さい場合は前方の、そうでない場合は後方の２分割点を探し、同様な操作を行ない、レコードを格納するブロックを探す。ロケーションテーブルが複数の連続領域から構成されている場合は、そのままではバイナリー・サーチが行なえないが、アドレス変換テーブルを利用して、あたかも連続領域であるかのように扱う事により、バイナリー・サーチを行なう事が可能となる。サブレンジに分割されている場合は、サブレンジ毎のロケーションテーブルの先頭もしくは最後の主キーの値をまず比較し、どのサブレンジが対象となるかを探す。その後、そのサブレンジに関して、上記と同様に、バイナリー・サーチを行い、該当するブロックを探し出す。

【００１３】ｎ番目のブロックに格納することになった場合以下のようになる。ブロックの中でレコードを挿入する位置を探す。レコードは主キーの順に並んでいるので挿入レコードより大きい主キーを持ったレコードの直前が挿入位置となる。まず、主キーの重複が無いかをチェックする。重複していた場合は格納できないのでエラーとして処理を行なう。重複していない場合、挿入レコードより後の位置にあるレコード（１つ以上）を挿入レコードが丁度入る大きさだけ後方にずらす。この際に、ずらしたレコードがブロックの中に収まればそれで処理は終了するが、納まらない場合は、オーバーフローブロックを１つ切り出して、プライマリーブロックからのポインター付けを行い、必要なだけオーバーフローブロックに格納する。その後、プライマリーブロックに挿入レコードと挿入レコードに続くレコードを格納する。プライマリーブロックとオーバーフローブロックの論理的な関係を図６に示す。

【００１４】既にオーバーフローブロックが存在する場合は、プライマリーブロックとオーバーフローブロックの双方を併せて格納が可能であればよい。また、オーバーフローブロックは一部分しか使用されずに、領域が有効に使用されない可能性がある。これを防ぐ為、複数のプライマリーブロックに対して１つのオーバーフローブロックを設ける事ができる。また、オーバーフローブロックの大きさは総て同一とするが、プライマリーブロックに比べて小さくすることが可能である。また、複数のプライマリーブロックから１つのオーバーフローブロックをポイントし使用するようにする事が可能である。この他に、オーバーフローしたレコードを、単独で格納領域に格納し、プライマリーブロックからポインター付けを行なう方法も採用する事が可能である。しかしながら、この方法はオーバーフローレコードが多い場合には、オーバーフローブロックを利用する場合に比較して、検索に時間がかかるというデメリットもあり、レコードの発生状況に応じた、格納方法の選択が必要である。

【００１５】次に、代替キーの格納、更新の場合の説明を行なう。代替キーテーブルは代替キーブロック中に代替キーの順番に並ぶように格納する。代替キーテーブルのエントリーは代替キー、そのキー値のレコードが格納されているブロックの物理アドレス、そのキー値のレコードの主キーからなる。代替キーテーブルはレコードの追加、更新に伴いエントリー数が変動するが、エントリーの増加に対しては、エントリーが挿入になる可能性が高く、追加型で発生する可能性は非常に少ない。この為、主キーと同様の管理ではうまく行なえない。最終格納予定レコード数に比較して、既に多数のレコードが存在する場合は、代替キーブロックにエントリーを格納する際に、一定の空き領域を設ける事により、挿入を効率的に処理できるが、初期のレコードの数が最終格納予定レコード数に比較して少ない場合は、キーの挿入によりオーバーフロー代替キーブロックが多数発生する事になる。この場合は、プレ代替キー・ブロックを使用する。プレ代替キーブロックは、代替キー・ブロックと同様な構造とし、ブロックの数は代替キーブロックの数のエントリーが格納できる大きさを、ブロックのサイズで割って求めた数とする。プレ代替キーブロックのエントリー数が、代替キーブロックの数と同じになった時点で、プレ代替キーブロックから代替キーブロックにエントリーを移動する。移動時には、代替キーブロックには原則として一つのエントリーを格納するが、同一の代替キーを持つエントリーは、同一の代替キーブロックに格納する。同一の代替キーを持つエントリーの数が多くて代替キーブロックに格納できない場合は、代替キー・オーバーフロー・ブロックを追加し格納する。例えば、最終格納予定レコード数を１００万件と想定しているとする。一つの代替キー・ブロックに１００エントリーの格納が可能とすると、代替キー・ブロックは１万個必要となる。エントリーが１万になるまで、プレ代替キー・ブロックに格納し、１万になった時点で、代替キー・ブロックにエントリーを移し替える。

【００１６】また、代替キーブロックの数が多く、プレ代替キーブロックを１段階設けた場合では、プレ代替キーブロックの数が多くなり、挿入が頻繁に発生し更新処理が非効率になる可能性がある。この場合は、プレ代替キーブロックを複数段階設ける様にする。前記の例で言えば、プレ代替キーブロックに１００００件のエントリーを格納する事になるが、１つのブロックに１００エントリーが格納可能なので、これを２段階とし、最初のプレ代替キー・ブロックは１００エントリーを管理し、エントリーが１００になった時点で、２段回目のプレ代替キーブロックに移し替えるという方法である。移し替えの具体例を図５に示す。この例は２段階のプレ代替キーブロックを設けた例である。

【００１７】次に、主キーでのレコード検索の方法を述べる。これは、レコードの挿入で、挿入位置を調べる為におこなった方法と同様に行なう。ここでは、挿入の例

と同様にバイナリー・サーチの例を記述する。まずロケーションテーブルの２分割点を探し、そのレコードが指しているプライマリーブロックにオーバーフローブロックがある場合にはオーバーフローブロックを含めて、そのブロック（当該ブロック）に格納されているレコードの主キーの値（以下、格納主キー値と略す）と挿入レコードの主キーの値（以下、挿入主キー値と略す）を比較する。挿入主キー値が格納主キー値の最小値より大きいか、当該ブロックの格納主キー値の最大値より大きく、次のブロックの格納主キー値の最小値より小さい場合は、当該ブロックに目的のレコードが存在するか、そのキー値のレコードはファイル上に存在しないかの何れかである。ブロックの中は主キーの順に並んでいるのでブロックを検索することにより、目的のレコードを検出するか、ファイル上に存在しないかを確認する事ができる。この判定に該当しない場合は、更に当該ブロックの主キー値と目的レコードの主キー値の大小を比較し、目的主キー値が小さい場合は前方の、そうでない場合は後方の２分割点を探し、同様な操作を行ない、レコードが格納されているブロックを探す。

【００１８】次に代替キーによる検索について説明する。代替キーでの検索は代替キーブロックを検索する。検索の方法としてはバイナリー・サーチが代表的であるが、これについては、主キーの検索で述べてあるので省略する。目的の代替キーが含まれる代替キーブロックを探し出す。更に代替キーブロックの中の目的の代替キーテーブルを探す。結果は、主キーの場合と同様にそのブロック内にエントリーが存在するか、その代替キー値をもつレコードはファイル中に存在せず、エントリーが無いかの何れかである。代替キーブロックに代替キー・オーバーフローブロックが存在するときには、その双方を検索の対象とする。

【００１９】目的のキー値を持つ代替キーテーブル（エントリー）が探し出せた場合は、そのエントリーにある物理ブロック番号から物理ブロックにアクセスし、そのブロック内でエントリー中にある主キー値と一致するレコードを探し出す。また、代替キーはノンユニークでいいので、代替キーブロック中の次のエントリーを見、代替キー値が同一である場合は、更にそのエントリーに対するレコードの検索を行ない、同一キー値を持つエントリーが無くなるまで実行する。

【００２０】次に、創生に関して説明する。創生とは、本方式のファイル上に既にレコードが複数存在しており、オーバーフローブロックが増加した等の理由で再創生する場合と、バックアップ媒体から本方式の適用媒体に展開する場合、それと、本方式で述べる格納方式以外の方法で格納されていたレコードを方法式に移行する場合が考えられるが、いずれも同様な方法で行なえる。再創生の場合は、ファイルを順次呼び出し方式にて主キーの順に読みだし、順編成ファイルを作成する。創生の場合も同様に順編成ファイルを作成する。

【００２１】次に、ロケーションテーブルと代替キーブロックを準備する。ロケーションテーブルの数は、格納を予定しているレコードの数を１ブロックに格納できるレコード数で割った数分を連続領域として確保する。代替キーブロックは代替キーの種類別に確保する。１つの種類の代替キーブロックは同一の大きさとし、その数は、次のように求める。１つの代替キーブロックに格納できるエントリー数（Ａ）を求める。格納予定レコード数をＡで割って求めた商が代替キーブロックの数となる。また、ブレ代替キー・ブロックを準備する。

【００２２】レコードは主キーの順に、ブロックに格納する。この際に予め挿入の頻度を予測または統計データから算出し、ブロック内に一定の割合の余裕を設けることができる。また、際間無くレコードを格納することも可能である。これは、挿入がどの程度行われるかで決定する。ブロック内の余裕はブロック毎に割合を変える事も可能である。格納に伴い、ロケーションテーブルへの記入も行なう。代替キーは まずレコードから代替キーエントリーを作成し、順編成ファイルに格納する。すべてのレコードの代替キーエントリーの作成が終了したら、代替キーの順にソートを行い、ソート後のエントリーを代替キーブロックに格納する。代替キーは代替キーブロックに格納するが、格納は、創生レコード数を最終レコード数で割った商を求め、その商に相当する割合で代替キーブロックに格納を行なう。これは、代替キーの発生が挿入型になる為である。

【００２３】代替キーの特殊な使用法も可能となる。現在のキーは、レコードの特定の位置に格納されているフィールドに対して、キーを設定するもので、例えば商品コードとか得意先コードといったように長さや形式が同一であった。本方式では、キーとレコードの関連付けが行なえれば良いので、文章データにキーを作る事が可能である。図７で示すように非定型なレコードにキーをつける事が可能となる。この例のように、フィールドの場所が一定せず、フィールドの長さも固定でないレコードに代替キーの設定が可能である。

【００２４】

【発明の効果】

①高速に主キーと代替キーによるランダムアクセスが行える。

②順次読み出し、書き込みも高速に行なえる。

③インデックスを使用せず、ロケーションテーブルと代替キーブロックを使用する為、データの追加変更削除に伴う変更が少なく、処理時間が短くレコードの格納が高速に行える。

④ロケーションテーブルや代替キーブロックの作成（創生）に関する時間がインデックス作成に比較して短時間ですむ。

④ロケーションテーブルや代替キーブロックの排他の範

囲が最小限で済み、デッドロックが起きにくい。
⑮予め、ブロックを作成する必要が無い。
⑯物理的な空きスペースが無くなるまで、ブロックの追加が行える。
⑰可変長レコードが扱え挿入も可能な為、レコード圧縮が可能（一般的には、圧縮レコードの更新ができないケースが多い）。
⑱代替キーも問題無く複数使用できる。
⑲データベースのメンテナンスの手間が大幅に削減できる。

【図面の簡単な説明】

【図１】ロケーションテーブル、最終ポインター構成。各々のバイト数は参考であり、この内容と異なる長さで構成は可能である。

＊【図２】ロケーションテーブルとブロックの関係。

【図３】代替キーテーブル（エントリー）と代替キーブロックの構成。バイト数は参考であり、この内容と異なる長さで構成は可能である。ＦＲＯＭ、ＴＯは当該ブロック中のキー値の最小値と最大値である。代替キーブロックのキー値の持ち方は、両方を持つ、または、いずれか一方のみでも可能である。

【図４】ブロックの構成。

【図５】プレ代替キーブロックの構成。

【図６】ブロックとオーバーフローブロックの関係図。解りやすくする為に、ブロックが連続領域にあるような図にしてあるが、実際には連続領域である必要は無い。

【図７】文章データにキーをつける例。

【図１】

ロケーションテーブル

ブロック#	物理アドレス
8BYTE	8BYTE

最終ポインター

ブロック#	物理アドレス	Pキー
8BYTE	8BYTE	任意長（最終ブロックの最大キー）

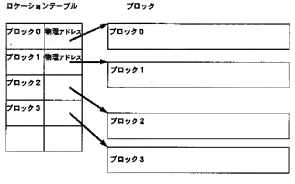
サブレンジ使用時　ロケーションテーブル

サブレンジ#	サブレンジ内ブロック#	物理アドレス
nBYTE	(8－n)BYTE	8BYTE

サブレンジ使用時　最終ポインター

サブレンジ#	サブレンジ内ブロック#	Pキー
nBYTE	(8－n)BYTE	任意長（最終ブロックの最大キー）
nBYTE	(8－n)BYTE	任意長（残除ブロックの最大キー）

【図２】

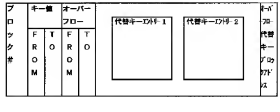


【図３】

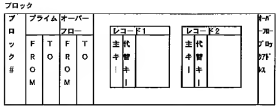
代替キーエントリー

ALキー	ブロック#	Pキー
任意長	8BYTE	任意長

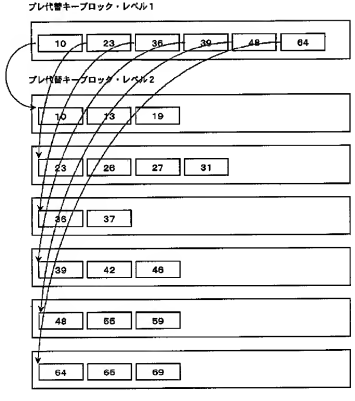
代替キーブロック



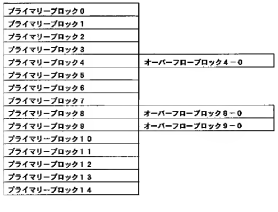
【図４】


ブロック
プロックヨ
プライム
FROM
TO
オーバーフロー
FROM
TO
レコードヨ1
主キー
代替キー
レコードヨ2
主キー
代替キー
ポインタブロック
XX

【図５】


プレ代替キーブロック・レベル１
10
23
36
39
48
64
プレ代替キーブロック・レベル２
10
13
19
23
26
27
31
36
37
39
42
46
48
55
59
64
65
69

【図６】


プライマリーブロック０
プライマリーブロック１
プライマリーブロック２
プライマリーブロック３
プライマリーブロック４
プライマリーブロック５
プライマリーブロック６
プライマリーブロック７
プライマリーブロック８
プライマリーブロック９
プライマリーブロック１０
プライマリーブロック１１
プライマリーブロック１２
プライマリーブロック１３
プライマリーブロック１４
オーバーフローブロック４－０
オーバーフローブロック８－０
オーバーフローブロック９－０

【図７】

文章データのキー

０００１　私は今日、学校へ行きます。
①　②　③　④

０００２　明日は、休みなので家でのんびりしようと思っています。
⑤　⑥　⑦ ⑧　⑨

上記の文章にキーをつける。
０００１、０００２を文章の主キーとする。ブロックは例れも１５５に種統されているとする。

代替キー表	ポイント付け（主キー値、ブロック番号、キーの先位）
のんびりしようと	０００２、１５５、８
家で	０００２、１５５、６
学校へ	０００１、１５５、１０
休みなので	０００２、１５５、４
行きます	０００１、１５５、１３
今日	０００１、１５５、２
思っています	０００２、１５５、１６
私は	０００１、１５５、０
明日は	０００２、１６５、０